ABSTRACT

CURRIE, KEVIN SCOTT. Factoring Large Numbers: Stealing Your Secrets. (Under the direction of Dr. E. L. Stitzinger.)

The purpose of this research has been to explore the methods and techniques currently used to factor large numbers. The RSA cryptosystem employs large numbers which are the product of two primes to encrypt and decrypt private messages. In order to break these codes, the first step is to factor the large public integer n into its two primes. Although there are many methods to factor these large integers, most are time consuming and may take decades or centuries to complete. The algorithms undertaken in this project are the predominate methods in use today and include the Pollard p-1 and the Quadratic Sieve. These methods are powerful and have the ability to factor large numbers. In order to accomplish these algorithms, the brute force method and the pseudoprime tests must be implemented and they are included in the research as well. The paper includes the methods for an intruder to steal the information sent over insecure lines of communication. In addition, it instructs the order in which the intruder should attempt to break the code, starting with the easiest methods first and then moving to more complicated and time consuming techniques.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	18.Apr.00	MAJOR REPORT

4. TITLE AND SUBTITLE
FACTORING LARGE NUMBERS: STEALING YOUR SECRETS

5. FUNDING NUMBERS

6. AUTHOR(S)
2D LT CURRIE KEVIN S

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NORTH CAROLINA STATE UNIVERSITY

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
12

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

BIOGRAPHY

Born in Chicago in 1976, 2nd Lt Kevin Currie graduated from Tinley Park High School in 1993. Next, he graduated from the United States Air Force Academy in 1998 where he became the first cadet in the history of the Academy to achieve a triple major with a minor. These include Mathematics, Operations Research, and Economics, and a minor in Russian. Additionally, Lt Currie was a member of the Wings Of Blue Parachute Team as an instructor, competitor, and demonstrator of precision skydiving. Following his education at North Carolina State University, Lt Currie will attend pilot training at Laughlin Air Force Base, Texas.

DEDICATION

For those who have been there: team, family, friends. . . thanks

"The good Christian should beware of mathematicians and all those who make empty prophecies. The danger already exists that mathematicians have made a covenant with the devil to darken the spirit and confine man in the bonds of Hell."

- St. Augustine

"Anyone who cannot cope with mathematics is not fully human. At best he is a tolerable subhuman who has learned to wear shoes, bathe and not make messes in the house. "

- Lazarus Long, "Time Enough for Love"

Factoring Large Numbers: Stealing Your Secrets

By

kevin currie

A thesis submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Masters of Science

MATHEMATICS

Raleigh

1999

APPROVED BY:

_____ _____

_____

Chair of Advisory Committee

TABLE OF CONTENTS

1. Introduction

With the creation of ENIAC, and the onset of the computer age, humans achieved the ability to add and multiply numbers quickly; unfortunately, it required an entire room to perform these operations. Currently, entering the twenty-first century, not only can computers handle large and complicated tasks, but they are found in the majority of American households. Connecting these computers, the internet provides a remarkable ability for people to transmit information rapidly.

This is seen in the recent increase of online banking. In 1995, the number of U.S. households banking online was 800,000; two years later, in 1997 the number rose to 4.5 million. With ease of use, households are beginning to prefer banking using the internet. Additionally, banks prefer this method; for a bank to process an in-person transaction, it requires an average cost of $1.07 while an internet transaction only costs $0.01. Online usage is not restricted to banking, retail sales steadily increase as more Americans shop using their computers. In 1997, electronic retail revenue amounted to $3.3 billion. One year later, that number jumped 60% to $5.3 billion. It is estimated that in America, 760 households join the internet every hour. Financial transactions over the internet will only increase in the coming years.

For this increase to take place, consumers must be confident that the information they send will be secure. As the internet transmits over

standard phone lines, the 1's and 0's sent are easily intercepted. This presents an interesting problem. How are these messages transmitted so that only the desired recipient will be able to read them? Are these encryption methods secure?

The study of cryptography creates methods to ensure the information sent will be secure and also attempts to break the codes. For any encryption method to be secure, it must be easier to create the technique than to break it. Currently, this is certainly true for the RSA technology. So the interesting aspect is: how is it broken?

2. RSA Encryption

RSA was published in 1978 by Rivest, Shamir, and Adleman, who also share the namesake. As a public-key cryptosystem, the encryption parameters are made public and are thus easily known to intruders. Yet this does not compromise the security of the transmissions. The first thing to know is how RSA works. The basic method is rather simple, the receiver chooses two distinct primes, p and q. The larger the values of p and q, the more secure the system becomes. Next, calculate $n = p*q$ and $m = (p-1)(q-1)$. Finally, an $a < m$ is chosen such that a and m are relatively prime and $b < m$ such that $a*b = 1 \bmod m$. The receiver then makes both a and n public and keeps b secure.

The sender converts the message text using some known mapping. Next, the sender raises the converted message to the power a and reduces modulo n. This is now sent over the insecure line of communication. The intended receiver raises the message to the power b and reduces modulo n. Now the receiver has the correct message – as $x^{ab} = x \bmod n$ – and no intruder can read the message without the decipher key b.

How does this cryptosystem keep the message safe? The key lies in the ability to easily generate large prime integers, while it is much more difficult to factor the product of those two large primes. The larger the primes, the harder to factor. As computers become faster and faster, the security of the system can be strengthened by simply increasing the size of the two primes.

So how do you factor the public integer n?

3. Factorization

As the integer n is public knowledge, and it is known that it is the product of two primes, it must be possible to find those two primes from the knowledge of n. If those primes are found, then determining b from the public a is relatively easy, and the system is compromised. Fortunately, given any integer, the fundamental theorem of arithmetic guarantees that the factorization into primes is unique up to order. Thus, given n, it can be factored completely and the intruder can steal the message.

Starting the process of factorization, the intruder uses the least complicated techniques first and then moves to more complicated methods. The first step is to use brute force. Next they check if the integer is prime, then moves to several other methods including the Pollard p-1 algorithm and the Quadratic Sieve.

3.1 Brute Force

The integer to be factored is either a product of primes or itself a prime. Simply dividing by all primes less than the integer will factor the number into its components. This method is guaranteed to work and find all the prime factors. Unfortunately, this method requires extraordinary time to complete since the number of primes less than an integer n is asymptotically $n/(\log n)$. For instance, the number of primes less than 10^9 is 50,847,478. Typically, the brute force is used to find any factors less than say 10,000. If none are found, more powerful algorithms are employed.

3.2 Primality Test

Coming to the next step, it is important to know the integer in question is not prime. This is required as the remaining tests assume the integer is not a prime. Thus, the following test based on Fermat's observation provides information on the primality of an integer. The

theorem is as follows: if **p** is a prime which does not divide **b**, then $b^{p-1} \equiv 1 \pmod{p}$. Unfortunately, there are composite integers **n** relatively prime to **b** such that $b^{n-1} \equiv 1 \pmod{n}$; these are said to be pseudoprimes for the base **b**. Testing these numbers with different bases may show the integers to be composite. Again, there are still composite integers which are pseudoprimes for all bases to which they are relatively prime. These numbers are called Carmichael numbers. In the interest of factorization, the integer **n** needs only to fail one of the several tests to insure it is not prime, and thus continue to the next step.

4. Pollard p-1

After factoring the small primes and ensuring the remaining integer is not prime, the next technique to employ is the Pollard p-1 algorithm. This method is also based on Fermat's Observation, and the theorem is as follows: if **p** is an odd prime then $2^{p-1} \equiv 1 \pmod{p}$. The Pollard p-1 algorithm assumes the integer **n** to be factored has a prime factor **p** with the property that the primes dividing **p**-1 are small. The restriction placed on **p** is such that **p**-1 divides 10000!. Then $\mathbf{m} = 2^{10000!}$ Mod **n** is computed. As **p**-1 divides 10000! then by the previous theorem, **m** is congruent to 1 modulo **p**, and thus **p** divides **m**-1. Therefore, there is a good chance that **n** does not divide **m**-1 and then **g** = gcd (**m**-1, **n**) will be a non-trivial divisor. Additionally, this

algorithm can be modified by changing the base, any number can be substituted for 2.

The maple program for the Pollard p-1 is as follows:

```
n := (insert number to be factored)
c := 2;
max := 10000;
m := c;
for i from 1 to max do
        m := m &^ i mod n:
        if (i mod 10) = 0 then
                g := gcd(m-1, n):
                if g > 1 then
                        g:
                        i := last + 2:
                fi;
        fi;
od;
g;
```

Although this algorithm is very powerful and can factor large integers, it has drawbacks. First, the gcd might equal n. In this case, the base c is changed to another number and the algorithm is implemented again. Also, if p-1 has only large factors, the algorithm might cycle forever. If p is the smallest prime dividing n, then the largest prime dividing p-1 is typically the number of cycles the Pollard p-1 algorithm requires. In the algorithm, *max* determines the number of cycles executed, with *max* set to 10000, the algorithm will usually find any prime factors which are less than two million. Increasing the cycles increases the size of the primes that can be found, but that will increase the running time of the algorithm as well.

5. Quadratic Sieve

One of the more powerful algorithms, the Quadratic Sieve is implemented after the small divisors have been found and the possibility of factoring with the Pollard p-1 has been depleted. As with the Pollard p-1, the integer in question must be composite, so the psuedoprime test must have been used to ensure the number is still composite.

Maurice Kraitchik realized that if random x and y were found such that $x^2 \equiv y^2 \pmod{n}$ then there is a 50-50 chance the gcd of n and x-y will be a nontrivial factor of n. Thus, the Quadratic Sieve attempts to find the suitable x and y. As this method is probabilistic, there is no guarantee that a factor will be found, but the technique is more likely to factor large integers quicker than other methods.

The first step in the Quadratic Sieve is to find a factor base and solve the congruencies $x^2 \equiv n \pmod{p}$ for each p in the factor base. To find the factor base, create the function, f(r) = r*r – n with domain r: k+1, k+2, . . . where k is the floor of the square root of n. The goal is to find the f(r)'s that factor into primes less than 10000. For a prime p less than 10000, p does not divide n – brute force was used and no factors less than 10000 were found – and if p divides f(r) then $n \equiv r^2 \pmod{p}$. This collection of primes used to divide f(r) is the factor base. To solve the quadratic congruencies, the following algorithm can be implemented:

```
INITIALIZE:        READ n, h, j, p
                   m ← n
                   v ← h
                   w ← (h*h – 2*m) mod p
                   CALL BINARY(j)
```

n *is known to be a quadratic residue* mod p. h *is chosen so that* $h^2 - 4n$ *is not a quadratic residue* mod p. j *is a positive integer, the last line converts* j *to binary notation.*

```
COMPUTE LOOP:      FOR k = t-1 to 1 BY –1 DO
                   x ← (v*w – h*m) MOD p
                   v ← (v*v – 2*m) MOD p
                   w ← (w*w – 2*n*m) MOD p
                   m ← m*m MOD p
                   IF b_k = 0 THEN  w ← x
                   ELSE DO
                           v ← x
                           m ← n*m MOD p
```

if v *is* v_k *and* w *is* v_{k+1}, *then the new value of* v *is* v_{2k}, *then new value of* w *is* v_{2k+2}, *and the new value of* x *is* v_{2k+1}. m *keeps track of the power of* n *modulo* p.

```
TERMINATE:         WRITE  v

BINARY(j):         i ← 0
                   WHILE  j > 0 DO
                           i ← i + 1
                           b_i ← j MOD 2
                           j ← ⌊j/2⌋
                   t ← i
                   RETURN
```

Return the values of t *and* b_i *to the caller*

Once the factor base is found and the quadratic congruencies have been solved, the next step is to perform the sieving operation to find sufficient f(r)'s which can be completely factored over the factor base.

Trying to factor 10000 integers over the factor base takes a considerable amount of time, with larger n's, the amount of integers increases. For example, David M. Bressoud recommends checking over 1 million r's if the n to be factored is only 66 digits. Therefore, implementing the sieve reduces the integers which are attempted to be factored.

For each f(r), if it is factorable over the factor base, then $f(r) = p_1^{a1} * p_2^{a2} * p_3^{a3} * \ldots$ And thus, $\log f(r) = a1 * \log p_1 + a2 * \log p_2 + a3 * \log p_3 + \ldots$ As the quadratic residues have already been solved, $n \equiv t^2$ or $(-t)^2 \pmod p$. Consequently, r is congruent to either t or -t modulo p, and thus p must divide f(r). Now, after finding the first r congruent to t modulo p, f(r) and every p^{th} f(r) thereafter is divisible by p. Additionally, the same is true for the first r congruent to -t and every p^{th} f(r). As the f(r)'s divisible by p are known without doing any division, the running time is shortened dramatically. Starting with a vector of zeros, add (log p) to the entry when p divides the corresponding f(r)'s. When the vector entry is close to log f(r), then it factors completely over the factor base. Calculating the log of each f(r) will take considerable time, instead, choosing an average value will save time. If the sieve is running over 2M values, then the logarithm of the absolute value of $(\lfloor \sqrt{n} \rfloor - M + i)^2 - n$ will be approximately TARGET = (log n)/2 + log M. After the sieving is done, there will be few entries which are sufficiently close to the TARGET that trial division over the factor base can be accomplished quickly. To be close enough, Robert D. Silverman

recommended setting CLOSENUF = TARGET - T*log(*pmax*) where *pmax* is the largest prime in the factor base and T is a constant near 2. Although this modification misses a few values of r for which $r^2 - n$ factors completely, the time saved more than compensates.

The last step in the Quadratic Sieve is to use Gaussian elimination to find a product of the f(r)'s which is a perfect square. For each integer which factors completely over the factor base, associate a binary vector the length of the factor base which has 1 as an entry if the corresponding prime appears as an odd power and a 0 if the prime is an even power. The matrix is formed by the collection of these vectors as rows adjoined with an identity matrix with as many columns as there are completely factorable integers. Perform Gaussian elimination until a vector is found such that there are zeros in each entry corresponding to a prime in the factor base. Each 1 in the remaining portion of the vector informs that the product of the corresponding f(r)'s is a perfect square. Therefore, multiplying the squared r's together will be congruent to the product of the factored f(r)'s (mod n). And thus, the two square terms x and y have been found. The next step is to find the gcd of x - y and n. About 50% of the time, the gcd is 1 or n and no new information is found, but with many entries in the matrix, it is likely that one will work.

With the Quadratic Sieve, large integers can be factored more efficiently than with brute force. The first and last of the three steps run quickly; it is the middle step, the sieve, where most of the computer time

will be spent. There are some modifications which may make the sieve faster. One such modification is to use a different f(r). Actually, the recommendation is to use several computers each with different f(r)'s checking a smaller range of r's.

6. Summation

Generating two large prime numbers is an easy task, factoring the product of those same two prime numbers is currently far more difficult. Thus, RSA keeps the secrets of the sender and receive safe. The above methods to factor the public key n still takes considerable time as n gets larger and larger. As computers become faster as well, the time required to perform the algorithms will decrease, keeping n constant. To overcome this, the information encrypted simply employs larger primes. But, this presents a problem if the message to be sent is intended to be kept secret for an extended period. An intruder who intercepts a message currently may not be able to decrypt the message, but given time and advances in computer technology, the message can certainly be cracked. Following the above methods, including brute force, Pollard p-1, and the Quadratic Sieve, the intruder is armed to steal. But it is the resourceful thief who will create new and better methods of factoring these large public integers, for two people cant keep a secret.

7. References

Bressoud, David M. *Factorization and Primality Testing.* New York: Springer-Verlag, 1988.

http://www.computerworld.com/home/Emmerce.nsf/All/rev BancAmerica Corp., Robertson, Stephens & Co. Feb. 23, 1998.

http://www.openmarket.com/intindex/99-05.htm The Internet Index, Number 24 Compiled by Win Treese. 31 May 1999

http://www.openmarket.com/intindex/99-02.htm The Internet Index, Number 23 Compiled by Win Treese. 28 February 1999.

http://www.computerworld.com/home/Emmerce.nsf/All/consumer Jupiter Communications, Inc., Feb. 23 1998.